# Pset1

## Hello:

En el archivo Hello.c se nos pide que hagamos una impresión en pantalla de un Hola, Mundo; en mi versión se hace una petición al usuario para ingresar el nombre de este y hacer la impresión de un saludo hacia el usuario, también incluye la impresión tradicional del Hola Mundo.

## Mario’s:

En los ejercicios sobre Mario se nos pide el elaborar una de las icónicas escaleras que se encuentran en los diversos mundos de dicho juego con el cambio de usar hashes “#” en lugar de bloques, en la versión Fácil se nos pide replicar una sola torre con un mínimo (1) y un máximo (8) de altura; para resolver esta versión use un Do while para obtener la altura deseada por el usuario y validar que la altura ingresada sea >1 y <8 seguido de esto entra un ciclo For donde este se ejecutara las misma cantidad de veces que el valor de altura, dentro de este For se encuentran 2 ciclos for más que sirven para empezar la impresión de los espacios y hashes, esta se hace de izquierda a derecha para que los hashes queden de tal forma que se represente escalera que se encuentra al final del nivel 1-1.

En la versión complicada de Mario se nos pide hacer un obstáculo con 2 escaleras en direcciones opuestas y un espacio entre ellas, para esto iniciamos con la misma validación de datos ingresados por el usuario usando un más Do while para obtener la altura deseada por el usuario y validar que la altura ingresada sea >=1 y <=8 seguido de esto entra un ciclo For donde este se ejecutara las misma cantidad de veces que el valor de altura, dentro de este for se encuentran 3 ciclos for mas donde el 1ero se encarga de imprimir los espacios de la escalera, el 2do imprime los hashes que forman la escalera entre medio de estos le sigue un printf que sirve para crear el espacio vacío entre cada escalera y el 3er ciclo for imprime la segunda escalera en dirección contraria a la primera seguido de un salto de línea para que se complete un ciclo.

## Cash:

En cash se nos pide crear una calculadora para saber exactamente cuantas monedas se le deben devolver al cliente por su cambio, en este se nos especifican 4 monedas a usar: Quarters = 25, Dimes = 10, Nickels = 5, Pennies = 1; dentro del código iniciamos creando la función get\_cents que nos sirve para solicitar al usuario la cantidad de dinero este se hace en un do while para verificar que la cantidad de dinero ingresada no sea menor que 0.

Seguido de esto está la función cal\_quarters, cal\_dimes, cal\_nickels y cal\_pennies que sirven para calcular la cantidad de monedas de 25, 10, 5 y 1 centavos respectivamente, el retorno de cada una de estas funciones será sumada y eso genera la cantidad de cambio en monedas que el usuario recibirá.

## Credit:

En credit.c se nos pide elaborar un archivo para verificación de tarjetas de crédito con ayuda del algoritmo Luhn, iniciamos pidiendo al usuario que introduzca el numero de su tarjeta de crédito, seguido a esto hacemos un conteo de la longitud de la tarjeta y verificamos si la longitud de la tarjeta queda en alguno de los estándares ya establecidos (13, 15 o 16), seguido de esto si la tarjeta pasa la verificación de longitud se pasa a dividir los números de la tarjeta para llegar a una suma total de los dígitos y empezar a usar la verificación de la tarjeta usando el algoritmo Luhn seguido a esto se hace un filtro con Ifs anidados para verificar el tipo de tarjeta que esta es.

# Pset2

## Caesar:

En caesar.c se nos pide implementar un programa que encripte mensajes usando el cifrado Caesar este criptosistema depende de que el usuario y el destinatario conocieran la clave secreta que seria el numero de lugares por los que el cifrado había desplazado cada una de sus letras; dentro del código se inicia haciendo una validación de argumentos donde el usuario debe seguir un orden de caracteres establecido (./caesar key), Seguido de eso realizamos una conversión a entero de la clave (key) y solicitamos al usuario que ingrese el mensaje a cifrar, el proceso de cifrado comienza en el ciclo principal del código donde dentro de un ciclo for y seguido de 2 condicionales if comenzamos a identificar y modificar cada una de las letras ingresadas por el usuario.

## Readability:

En Readability.c se nos pide implementar un programa que calcule el nivel de grado necesario para comprender un texto según Scholastic; se nos da la siguiente formula para realizar el calculo index = 0.0588 \* L – 0.296 \* S – 15.8 donde L es el numero promedio de letras por cada 100 palabras en un texto y S es el numero promedio de oraciones para cada 100 palabras en un texto con esto dentro del código iniciamos pidiéndole al usuario que ingrese un texto seguido de eso entra a un for que realiza el conteo de letras de cada palabra dentro del texto, saltando a otro for que realiza el conteo de cada palabra dentro del texto y terminando en un 3er for que cuenta cada oración dentro del texto, terminando con los ciclos se procede a realizar el calculo dado por Sholastic y realizar un redondeo para luego indicar el nivel de lectura que le corresponde a cada texto.

## Substitution:

En substitution.c se nos pide crear un programa que implemente un cifrado de sustitución es una forma de cifrado reversible en un mensaje reemplazando una letra por otra para esto se requiere una clave que en este caso sería un mapea de cada una de las letras del alfabeto; en código realizamos la misma verificación que se hizo en caesar.c verificando los argumentos requeridos seguido de eso se declara la variable donde se almacenara la longitud y procedemos a validar que la longitud llegue hasta 26, luego de eso verificamos dentro de un ciclo for que la clave(key) sea del alfabeto y que no se repita, después le pedimos al usuario que ingrese el texto a cifrar y dentro de un ciclo for se comienza a sustituir cada una de las letras dentro del texto ingresado.

# Pset3

## Plurality:

En plurality.c se nos pide crear un programa que corra un sistema electoral mayoritario donde el candidato con mayor número de votos será el ganador dentro del código verificamos que los argumentos concuerden con los solicitados, seguido de eso verificamos la cantidad de candidatos inscritos y que estos sean menos de 9, seguido a eso se registra el voto y el nombre del candidato al que se ha votado luego se verifica si los datos registrados concuerdan con los candidatos inscritos y así se confirma el voto del usuario; en la función print\_winner se hace la verificación de votos y se elige el candidato ganador.

## Runoff:

En Runoff.c se nos pide implementar un programa que corra un sistema electoral preferencial, un programa que realice un conteo para realizar la premiación del candidato con la mayoría de votos, cada votante debe elegir su top 3 de candidatos en las elecciones, se nos pide rellenar la función vote donde se itera sobre la cantidad de candidatos con un ciclo for, dentro de este se verifica si el candidato electo es igual que las opciones dadas en la estructura y si es esta dentro de la estructura guardara la preferencia del votante y su posición; luego de eso nos piden completar la función print\_winner este itera en la cantidad de votos con un ciclo for y verifica si algún candidato tiene mas de la mitad de los votos con un condicional if y cumple con la condición imprime el nombre del candidato y retorna true; la función find\_min se encarga en buscar si el candidato a sido eliminado y asigna a MAX\_VOTER la cantidad de votos del candidato; la función is\_tie verifica si el candidato no ha sido eliminado y si los candidatos no son iguales al min y la función eliminate se encarga en verificar si el candidato es igual a min y si lo es lo elimina.

## Tideman:

En tideman.c se nos pide implementar un programa que corra un sistema de elecciones Tideman dentro del código se nos pide completar la función vote dentro de la función se incluye un ciclo for que incluye un if donde se verifica si el nombre del candidato es válido y si lo es actualiza la preferencia.

Luego nos da la función record\_preferences que se encarga en recordar las preferencias dentro de este hay 2 ciclos for que se encargar en recorrer cada una de las casillas del arreglo.

Luego nos da la función add\_pairs donde hay 2 for anidados que contienen un condicional if el primero verifica si la preferencia de candidato es mayor que el otro si lo es entra y determina el ganador y el perdedor en el else if se hace lo contrario.

Luego sort\_pairs ordena los pares en orden decreciente por el orden de victoria dentro del código están 2 for anidados que tienen dentro un condicional if que ordena los elementos dentro del arreglo.

Luego la función lock\_pairs que debe crear el grafico locked itera sobre los pares con un ciclo for que contiene un condicional if donde si la función cycle retorna false bloquea todos los pares.

Luego la función print\_winner con for e ifs anidados crea un ciclo para imprimir los candidatos según el orden en el que estén dentro del arreglo.

# Pset4

## Filter fácil:

En Filter nos piden modificar las funciones grayscale, sepia y reflect para cambiar la foto que nos brindan en los archivos iterando las dimensiones de la imagen cambiando el valor de los pixeles.

## Filter Dificil:

En la función Grayscale contiene 2 for anidados que iteran sobre las dimensiones de la imagen y dentro de ellos calcula el promedio del valor del pixel y establece pixeles; con la función reflect contiene 2 for anidados que dentro de los se guardan la imagen original temporalmente dentro de los arreglos y luego crea 2 copias de izquierda y derecha de la foto y la almacena temporalmente; la función Blur crea una imagen temporaria donde con un for que recorre los pixeles vecinos de la imagen u verifica si son validos o no luego almacena el valor de la imagen en arreglos y seguido de eso hace el promedio de los pixeles vecinos luego copia la imagen con blur en la original, siguiente este la función edges crea un arreglo temporal para la imagen luego inicializa los arreglos de Sobel luego con 2 for anidados itera sobre columnas y filas dentro de dichos for se encuentran 2 pares mas que para cada pixel iteran vertical y horizontalmente dentro de ellos se verifica si el pixel esta fuera de las filas y si están dentro de las columnas luego calcula el algoritmo sobel para luego asignar los valores a los pixeles.

## Recover:

En recover.c nos piden implementar un programa que recupere archivos JPEG de una imagen forense en código iniciamos con una apertura de archivos y comprobamos que el archivo existe y entra la condicional while por la condición de buffer dentro de este se encuentran 2 ifs anidados donde se comprueba que la imagen este abierta e imprime en el arreglo archivo en caso de encontrar un JPG escribe sobre la imagen

# Pset5

## Speller:

En speller nos piden implementar un programa que verifique la ortografía de un archivo, nos piden modificar ciertas funciones como check donde se crea la hashtable en la palabra original y procede a acceder a la lista con el índice del hash table, seguido de eso en la función load se hace la apertura del archivo diccionario luego creando un arreglo para la longitud de la palabra y creando un ciclo para leer caracteres del string con un condicional while que contiene la creación de un nuevo nodo para la palabra y crea una copia en el nuevo nodo seguido a esto aplicando la hashtable en la palabra siguiente asignando el nodo siguiente el puntero de arreglo de hashtable.

# Labs

## Population:

En population.c nos piden determinar cuanto tiempo le toma una población alcanzar un tamaño en específico en este código pedimos al usuario que ingrese la cantidad inicial y validamos que esta cantidad sea menor que 9 con una condicional do while, seguido pedimos al usuario que ingrese a la cantidad que desea que llegue la población en un punto final y validamos que dicha cantidad no sea menor que la cantidad inicial ingresada; luego de eso ponemos en practica la formula que se nos dio para calcular los años que tomaría llegar a la cantidad esperada.

## Scrabble:

En scrabble.c nos piden determinar cual de estas palabras valen mas en Scrabble en este nos dan arreglos ya establecidos con la puntuación y los ascii de cada letra del alfabeto luego dentro de la función compute\_score verificamos si la letra es mayúscula o minúscula, si es capital y busca dentro del arreglo ascii luego toma su índice y le asigna puntos según este.

## Sort:

Nos pide analizar tres programas de ordenamiento para determinar que algoritmo usan midiendo el tiempo en el que estos tardan en completar la tarea

## Volumen:

Nos piden escribir un programa que modifique el volumen de un archivo de audio

## Inheritance:

Nos piden simular una herencia de tipos de sangre para cada miembro de una familia para esto nos dan una estructura con la que debemos crear la herencia de tipo de sangre en la familia